ESOC Navigation Support Office
H. Boomkamp

Software modernization in support of LEO and Multi-Constellation Processing

INTRODUCTION

Over the past decades, few areas of science and engineering have known more rapid progress than information technology. Nonetheless, the many improvements in software engineering seem to be largely ignored by the other scientific disciplines. This is especially remarkable in cases where productivity depends directly on the capacity and adaptability of large software systems, as is the case for the IGS Analysis Centres.

By the time of the 2006 IGS Workshop, routine IGS processing is still largely based on FORTRAN 77 technology, which was already quite obsolete when IGS started operations: by the early 1990's, the IT industry was generally abandoning the idea of developing large software systems in FORTRAN, ALGOL or COBOL, in favour of more compact and flexible third generation languages, exploiting concepts like Object Oriented Programming and Pervasive Information Management.

Such innovations in Information Technology are usually driven by economic arguments, like reduced development trajectories or improved reusability of existing source code. These arguments are equally valid for the IGS Analysis Centres, which typically have limited budgets for software developments. However, there are two further reasons why software modernization is of critical importance to IGS.

The first reason is the strong increase in processing capacity that is needed to cope with multiple GNSS constellations (GPS, GLONASS, Galileo), larger station networks, the arrival of Low Earth Orbiters (higher data rates for the entire solution), and the drive towards shorter IGS product latencies. Modern methods are available for optimizing software systems in terms of memory and CPU, but it is typically impossible to apply such methods on complex conglomerates of existing FORTRAN code.

The second reason is that due to continuous changes and increasingly complex software, the maintenance process of a large FORTRAN system becomes saturated: the development trajectories become too long to cope with the quickly changing requirements, so that the system that is actually available for operational use lags more and more behind with the state of art in knowledge.

At ESOC IGS Analysis Centre, the FORTRAN system that is used operationally shows the signs of both sub-optimal performance and saturated maintenance. For this reason, a new C++ system "ROBOD" has been developed around various optimization methods that maximize processing capacity within available hardware, while strongly reducing overall code size and code complexity.

This poster reports on the status of the ROBOD development and compares its performance against the operational ESOC IGS software.

ROBOD design drivers

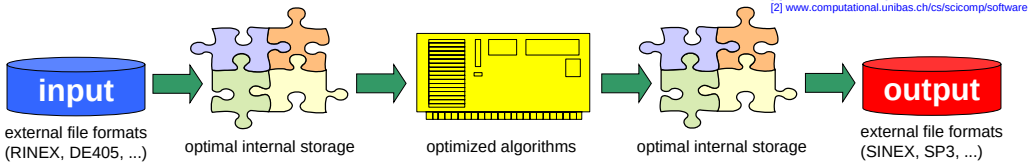
- minimized use of memory
- minimized use of CPU
- minimum complexity for user
- minimum code complexity

The Relational Object Base for Orbit Determination consists of interacting sets of dynamically allocated C++ objects. The data properties of these objects are equivalent to data fields of records in a relational database, and sets of similar objects (satellites, stations, observations, ...) correspond to record tables in a relational database. Some data properties may be pointers to objects in other sets, corresponding to index numbers to records in other tables of a database.

This exact mapping between a network of C++ objects and a relational database allows all internal data to be managed by generic database principles. The most important of these is *Uniqueness of Information*: no data is ever repeated in more than one record. Instead, different records may use relations (pointers) to a same data record (object) in another table, which holds the *unique* information of interest. The design of all class hierarchies is driven by the principle of uniqueness.

In a GPS parameter estimation process, the use of CPU is dominated by the construction and inversion of the normal equations, for which the ROBOD system uses the ATLAS [1] libraries in combination with the Pardiso [2] solver, allowing efficient parallelization over various CPUs. All lower level data processing steps, like the computation of observation residuals or the integration of satellite orbits are handled in a distributed way by the individual objects.

[1] math-atlas.sourceforge.net
[2] www.computational.unibas.ch/cs/scicomp/software



Visualization of BAHN and ROBOD runs

Summary of test case that was executed by both systems

- **Network of 60 IGS stations**
- **Constellation with 29 GPS satellites**
- **12 hour solution arc**
- **5 minute data samples of undifferenced iono-free GPS data**

Estimated parameters

- ***Initial position and velocity of satellites***
- ***6 SRP parameters per satellite***
- ***Epoch-dependent clocks***
- ***Station positions***
- ***ambiguity bias per pass***

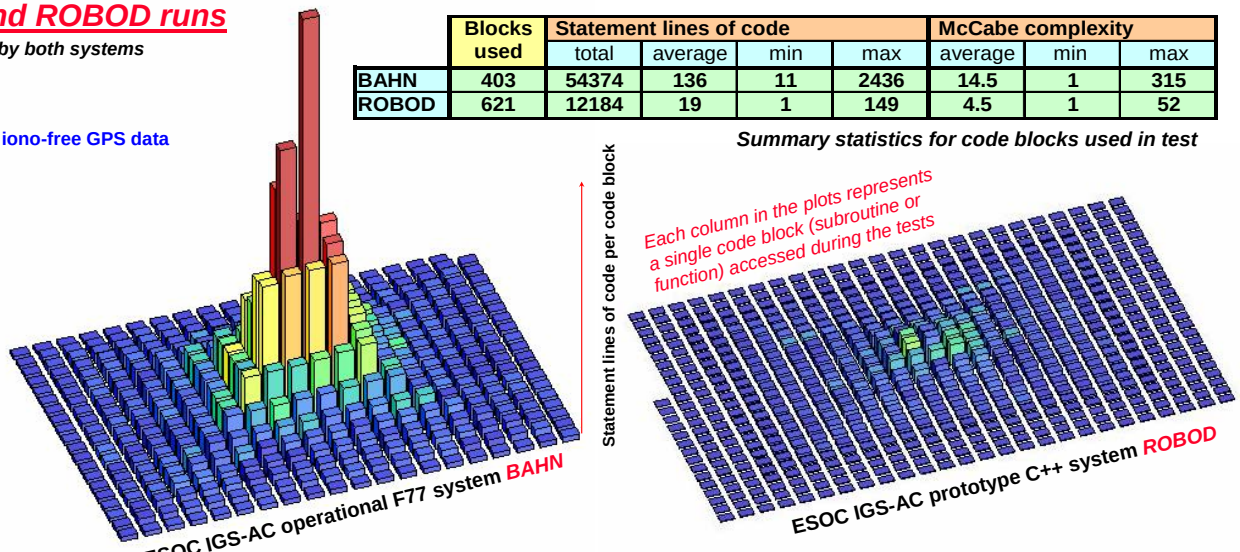
The Figures on the right visualize the substantial reduction in code size and code complexity that can be achieved by using state of art information technology

SLOC - Statement lines of code
Source code metric that indicates the size of a subroutine or function.

McCabe Cyclometric Complexity
Source code metric that defines the number of independent paths through a code block, defining testability and maintainability of the source code

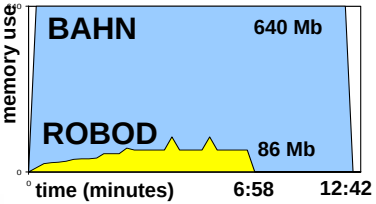
	Blocks used	Statement lines of code				McCabe complexity		
		total	average	min	max	average	min	max
BAHN	403	54374	136	11	2436	14.5	1	315
ROBOD	621	12184	19	1	149	4.5	1	52

Summary statistics for code blocks used in test



McCabe cyclometric complexity per code block

Use of hardware by the two processes



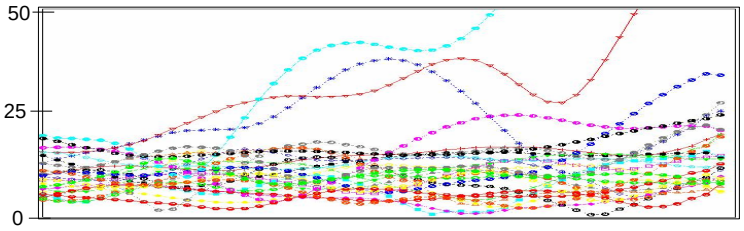
The Figures demonstrate the typical advantages of software modernization:

- ***Code size reduced by factor > 4***
- ***Code complexity reduced by factor > 3***
- ***Hardware efficiency improved by > 7***

The immediate benefits are:

- ***Shorter development trajectories***
- ***Shorter familiarization periods***
- ***Cheaper development and maintenance***
- ***Easier / faster adaptation to new tasks***
- ***Lower bug probability***
- ***Easier configuration control***
- ***Less documentation to maintain***
- ***etc., etc. !***

Current ROBOD orbit precision: differences (cm) with IGS final for test arc



Summary and conclusions

The ROBOD system demonstrates what the world of Information Technology has known for several decades, namely that large applications like GPS parameter estimation software can be developed in far simpler and more efficient ways than via traditional, large FORTRAN systems. The latter tend to suffer from extreme code complexity, poor internal data organization, limited flexibility towards change and steady degradation of code structures. The fatal consequence for an IGS Analysis Centre is a complete saturation of the available software maintenance trajectories, and therefore an ever increasing gap between desired capabilities and capabilities that are actually operationally available.

The total workload spent so far on the ROBOD development accumulates to about 15 months of fulltime work for one person, which is only a very modest fraction of the accumulated effort that has been invested in the ESOC FORTRAN systems over the years. The most relevant "innovations" that the ROBOD software exploits are Object Oriented Programming in C++, and Uniqueness of Information, as known from relational databases. Both these concepts have in fact been available for at least two decades, which implies that a system like ROBOD could have been developed long before IGS started its operations.

The main benefits of these design features are

- maximized processing capacity within the limits of the available hardware
- far shorter development trajectories for future modifications (eliminating any form of maintenance saturation)
- much shorter familiarization trajectories both for users and developers of the system.

Such advantages will greatly improve the productivity of the ESOC IGS Analysis Centre, and will be of particular interest to demanding future applications like reprocessing of historic data, multi-constellation solutions, and high-rate solutions involving GNSS and LEO satellites.

Completion of the system will require the equivalent of several further months of work, which will however be spread out over a longer period due to other obligations. Nonetheless, the current state of the system is mature enough to demonstrate the substantial benefits of software modernization, and to suggest that other IGS Analysis Centres may benefit equally from a radical upgrade of their processing systems. In general, the effort of doing so seems to be grossly over-estimated, while the long-term benefits of software modernization are either overlooked, or seriously undervalued.

... measure your own software!

All code metrics for the tests presented on this poster have been generated with
CCCC for C++,
available at http://cccc.sourceforge.net
and
Understand for FORTRAN
available at http://www.scitools.com/uf.html

esa

IGS Workshop 2006 ESOC